REMARKS

The Specification has been amended for clarification purposes only and does not present new matter. Claims 1, 2, 8, 10, 12, 14, and 17 have been amended. Claims 5-7, 9, 13, 15, and 16 have been canceled, and claims 18-20 have been added. Thus, claims 1-4, 8, 10-12, 14, and 17-20 are currently pending in the case. Further examination and reconsideration of the presently claimed application are respectfully requested.

Section 103 Rejections

Claims 1-17 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,327,529 to Fults et al. (hereinafter "Fults") in view of U.S. Patent No. 6,434,694 to Slaughter et al. (hereinafter "Slaughter"), and in further view of pages 145 to 159 of a publication entitled "Platform Performance: Strategies and Tactics," written by Steve Wilson (hereinafter "Wilson"). To establish a *prima facie* obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974), MPEP 2143.03. Obviousness cannot be established by combining or modifying the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion or incentive to do so. *In re Bond*, 910 F. 2d 81, 834, 15 USPQ2d 1566, 1568 (Fed. Cir. 1990). The cited art does not teach or suggest all limitations of the currently pending claims, some distinctive limitations of which are set forth in more detail below.

None of the cited art teaches or suggests a display system including: (i) a platform-independent application program, (ii) a platform-independent software component for fetching list data from memory for producing a display image of the list data upon a display without invoking a platform-dependent display routine, and (iii) a platform-independent peer component for intercepting invocations of platform-dependent display routines from said platform-independent application program and for routing the intercepted invocations to said platform-independent software component. Amended independent claim 1 recites in part:

> A display system, comprising ... platform-independent application program containing invocations of platform-dependent display routines ... a platform-independent software component for fetching list data from the memory for producing a display image of the list data upon the display without invoking a platform-dependent display routine; and a platform-independent peer component coupled between the platform-independent software component and the platform-independent application program, for intercepting

said invocation of platform-dependent display routines and for routing the intercepted invocations to said platform-independent software component.

Support for the amendments to claim 1 may be found in the Specification, for example, on page 30, line 4 to page 31, line 15.

The presently claimed case provides a display system, method and computer-readable storage device for overcoming one of the many problems that occurs when programmers attempt to mix AWT components (i.e., Java) and Swing components within a Java application program. In general, the presently claimed case provides a system of software components – referred to as "AWTSwing" components – that enable a Swing listbox or choice control to be substituted for an AWT listbox or choice control. However, direct substitution of Swing and AWT components presents many problems (*see*, e.g., Specification, page 22, lines 1-28), and as such, is often avoided by conventional programmers.

In order to substitute a Swing component for an AWT component, the presently claimed display system provides a platform-independent peer component. When an AWT listbox or choice control is displayed by an application program (e.g., a Java application) containing invocations (or call routines) of platform-dependent display routines, a platform-independent peer component (e.g., List Peer 156, FIG. 12) is created for intercepting and routing the call routines from the AWT control (e.g., List Component 144, FIG. 12) to a corresponding Swing control (e.g., Swing JList 160, FIG. 12). This is beneficial because, unlike AWT listbox/choice controls, Swing listbox/choice controls do not create a redundant copy of the list originally stored in memory, thereby saving both time and memory space.

Fults discloses a process of designing user interfaces for applications programs (Fults, Title). In particular, the GEOS process disclosed by Fults involves "creat[ing] a generic user interface description (in the form of objects with attributes and hints)..." (Fults, column 15, lines 62-67). The "GEOS [operating] system reads the [generic] description, interprets it, and produces a realization of the program user interface which visually and behaviorally conforms to the explicit and implicit guidelines of a particular style guide." (Fults, column 16, lines 10-15). In other words, Fults discloses the creation of a platform-independent user interface, which when run by a particular operating system, conforms to the guidelines of the platform-dependent style guide (i.e., display routine) associated with that operating system. When displaying various "gadgets" or components (e.g., windows, menus, buttons, scrolling lists, etc.), "the operating system software... handles the details of actually selecting, arranging and otherwise managing the gadgets used to represent the component." (Fults, column 25, lines 6-10). In other words, the platform-

independent user interface of Fults invokes a platform-dependent display routine from the operating system to produce images of the gadgets (much like AWT). Accordingly, Fults cannot provide teaching or suggestion for the platform-independent software component or the platform-independent peer component, as recited in present claim 1. Consequently, Fults fails to teach or suggest all limitations of present claim 1.

On page 2 of the Office Action, the Examiner admits that Fults fails to provide teaching for the presently claimed software component, but suggests that Slaughter "teaches the creation of a list, [and] further teaches the display not dependent on invoking a display routine from the operating system (see column 6, lines 61 through column 7, line 6)." The Applicant concedes that Slaughter may provide some teaching for "the creation of a list", as suggested in the above-mentioned Office Action statement. As noted below, for example, Slaughter discloses a bus manager that "creates an array of MemoryDescriptor objects" in the passage cited in the Office Action. However, merely creating a list (e.g., an array of MemoryDescriptor objects) is not a claimed aspect of present (or original) claim 1, and therefore, irrelevant to the patentability of that claim.

For at least the reasons set forth in more detail below, the Applicant respectfully disagrees with the Examiner's assertion that Slaughter teaches a means for displaying an image of the list upon a display without invoking a (platform-dependent) display routine from the operating system.

Slaughter discloses a computer-implemented method for allocating memory resources to a platform-independent device driver (Slaughter, Abstract). However, there is absolutely no mention within Slaughter of a software component for fetching list data from memory for producing a display image of the list data upon a display without invoking a platform-dependent display routine. Slaughter also fails to mention a platform-independent peer component for intercepting invocations of platform-dependent display routines from a platform-independent application program and for routing the intercepted invocations to said platform-independent software component. Instead, and as recited in the passage cited by the Examiner, Slaughter discloses a bus manager that "creates an array of MemoryDescriptor objects that may be used by the device driver." (Slaughter, column 6, line 67 to column 7, line 1). Even if the "array of MemoryDescriptor objects" is considered equivalent to the presently claimed "list data", Slaughter still fails to disclose a software component, which fetches the "array of MemoryDescriptor objects" from memory for producing a display image of the list data upon a display. Slaughter simply does not teach or suggest that the "array of MemoryDescriptor objects" are displayed (or could be displayed) as

an image on a display, nor does Slaughter provide any motivation for doing so (particularly since the invention of Slaughter concerns methods for allocating memory resources to device drivers, and does not address a means for producing images on a display). In the same manner, Slaughter does not and cannot provide teaching or suggestion for a platform-independent peer component for intercepting invocations of platform-dependent display routines from a platform-independent application program and for routing the intercepted invocations to the platform-independent software component. Consequently, Slaughter does not teach or suggest all limitations of present claim 1.

Further statements in the Office Action suggest that "Wilson teaches Swing which is Java's version of an OS independent user interface display API, similar to that of Slaughters OS independent control... It would have been obvious to one of ordinary skill in the art, having the teachings of Fults, Slaughter, and Wilson before him at the time the invention was made to modify the list display of Fults to include the independent display routines of Slaughter and Wilson." (Office Action pages 2-3). The Applicant respectfully disagrees, for at least the reasons set forth in more detail below.

First of all, Slaughter does not teach or suggest an "independent display routine", as alleged in the above Office Action statement, and therefore, cannot be combined with Fults to teach or suggest the presently claimed platform-independent software component. Pointing to a "platform-independent device driver" within the teachings of Slaughter does not provide support for a platform-independent software component for fetching list data from memory for producing a display image of the list data upon a display without invoking a platform-dependent display routine. Since neither Fults nor Slaughter disclose the presently claimed software component or the presently claimed peer component, Fults and Slaughter cannot be combined to do so.

In addition, Wilson cannot be combined with Fults (or Slaughter) to overcome the deficiencies therein. In particular, the "list display" of Fults cannot be modified with the teachings of Wilson, since neither Fults nor Wilson suggest the desirability for doing so. The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990); MPEP 2143.01.

As noted above, Fults discloses a generic user interface, which when run by a particular operating system, conforms to the guidelines of a platform-dependent style guide (i.e., display routine) associated with that operating system. Wilson, on the other hand, provides a brief overview of Swing Models and Renderers, which are known to be platform-independent. Though Swing may also be considered to provide a "generic user interface", one simply cannot substitute the platform-independent Swing components of Wilson for the generic user interface of Fults. In other words, Fults provides no desirability for incorporating Swing components within his generic user interface, especially since Swing components do not conform to the guidelines of platform-dependent style guides (i.e., display routines) associated with the operating system running the generic user interface.

None of the cited art teaches or suggests a computer-readable storage device or method for generating a display image, which includes intercepting platform-dependent invocations by a first platform-independent software component, and generating a display image using a second platform-independent software component to generate a display list image without creating a copy of list data stored by an application program. Amended independent claim 12 recites: in part:

> A method for generating a display image, comprising: running a platform-independent application program ... said platform-independent application program invoking platform-dependent image display software for generating a display image; intercepting said platform-dependent invocations by a first platform-independent software component; and generating a display image using a second platform-independent software ... wherein said second platform-independent software component generates a display list image without creating a copy of list data stored by said application program.

Amended independent claim 17 recites a similar limitation.

Neither Fults, Slaughter nor Wilson provide teaching or suggestion for intercepting platform-dependent invocations by a first platform-independent software component, and using a second platform-independent software component to generate a display list image without creating a copy of list data stored by an application program. Instead, Fults teaches a contradictory method by first creating a list file in a generic (i.e., platform-independent) GenListClass object (*see*, column 10, lines 12-26 of Fults), which may be later read and interpreted by platform-dependent user interface software (*see*, column 15, line 62 to column 16, line 16 of Fults). Given the teachings of Fults, one simply cannot assume that the disclosed method may be performed in reverse. In other words, Fults provides no teaching, suggestion, or motivation for intercepting platform-dependent invocations (i.e., call routines or display instructions) and transferring said invocations to a second platform-independent software component (that draws from a

library of platform-independent commands) for generating the display list image, as recited in present claims 12 and 17. Consequently, Fults does not teach or suggest all limitations of present claims 12 and 17.

Slaughter doesn't even teach a method for generating a display image. Though Slaughter may teach "the creation of a list," as suggested by the Office Action, Slaughter definitely fails to teach or suggest the presently claimed method step of intercepting platform-dependent invocations and using a second platform-independent software component (that draws from a library of platform-independent commands) for generating the display list image without creating a copy of list data stored by an application program. Consequently, Slaughter does not teach or suggest all limitations of present claims 12 and 17.

Finally, though Wilson discloses platform-independent commands (i.e., Swing components), and even mentions the Swing JList component, Wilson does not disclose the presently claimed method step of intercepting platform-dependent invocations and using a second platform-independent software component (that draws from a library of platform-independent commands) for generating the display list image without creating a copy of list data stored by an application program. The Applicants assert that Swing components do not inherently include a means for redirecting or transferring a platform-dependent invocations or call routines (such as call routines to AWT components) to a library of Swing commands. As a result, Slaughter cannot be relied upon to teach or suggest all limitations of present claims 12 and 17.

Since none of the cited art teaches or suggests the aforementioned limitation, the cited art cannot be combined in such a manner that teaches or suggests the aforementioned limitation of the presently claimed case.

For at least the reasons set forth above, none of the cited art, either separately or in combination provides motivation to teach or suggest all limitations of present claims 1, 12, and 17. Therefore, claims 1, 12, and 17, as well as claims dependent therefrom, are patentably distinct over the cited art. Accordingly, removal of this rejection is respectfully requested.

Patentability of Added Claims

The present Amendment adds dependent claims 18-20. Claims 18-20 are dependent on independent claim 17; therefore, they are patentably distinct over the cited art for at least the same reasons as claim 17. Accordingly, approval of added claims 18-20 is respectfully requested.

CONCLUSION

This response constitutes a complete response to all issues raised in the Office Action mailed March 15, 2004. In view of the remarks traversing rejections, Applicants assert that pending claims 1-4, 8, 10-12, 14, and 17-20 are in condition for allowance. If the Examiner has any questions, comments, or suggestions, the undersigned attorney earnestly requests a telephone conference.

No fees are required for filing this amendment; however, the Commissioner is authorized to charge any additional fees which may be required, or credit any overpayment, to Conley Rose, P.C. Deposit Account No. 03-2769/5468-07400.

Respectfully submitted,

Kevin L. Daffer
Reg. No. 34,146
Attorney for Applicant(s)

Conley Rose, P.C.
P.O. Box 684908
Austin, TX 78768-4908
Ph: (512) 476-1400
Date: June 15, 2004
JMF